

REMARKS

The Office Action has been carefully considered. Claims 31 and 35-37 were rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent 5,491,808 to Geist, Jr. ("Geist Jr."). Claims 1-9, 11-16, 27-30, 32-34, and 39-53 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Geist, Jr. in view of U.S. Patent 5,689,707 to Donnelly ("Donnelly"). Claims 10 and 38 were rejected under U.S.C. § 103(a) as being unpatentable over Geist, Jr. in view of U.S. Patent 5,689,707 to Donnelly ("Donnelly") and further in view of PCT WO 95/22104 to Parker et al. ("Parker"). Applicants respectfully traverse these rejections.

By present amendment, claims 27, 31 and 32 have been amended for clarification and not in view of the prior art. Applicants submit that the claims as filed were patentable over the prior art of record, and that the amendments herein are for purposes of clarifying the claims and/or for expediting allowance of the claims and not for reasons related to patentability. Reconsideration is respectfully requested.

Applicants thank the Examiner for the interview held (by telephone) on May 26, 2004. During the interview, the Examiner and applicants' attorney discussed the claims with respect to the prior art. The essence of applicants' position is incorporated in the remarks below.

Applicants' technique is generally directed to a system and method for testing kernel mode components in an operating system. Tests for specific errors are identified to the kernel at the time of loading the component. Kernel system calls by the component are re-vectored to a verifier component which controls execution of the system calls and

tests for the specific errors identified for that component. For example, if the type of error notified was memory errors, then the verifier component uses a series of memory techniques such as isolating allocated memory, testing memory usage, and marking deallocated memory. The verifier also checks that all memory was deallocated when the component is unloaded to detect memory leaks. Additionally, the verifier checks that any requested kernel resources are deleted when the component unloads. These resources include queues, lookaside lists, worker threads, timers and other resources. The component verifier may also be configured to simulate low resource conditions to discover improper system calls such as raising interrupt request level to access invalidated paged code or data.

Geist Jr., on the other hand, relates to tracking memory allocation and deallocation for an application program in a network file server. The technique uses an allocation tracker to record memory allocation by user programs and exception messages for memory requests. Unlike applicant's verifier that operates in kernel mode, Geist Jr. describes an application monitor that operates in user memory space. It overcomes the limitation of network loaded modules (NLM) that do not identify individual memory allocations, deallocations, and reallocations as they occur in the running of a particular NLM. The network operating system makes a set of APIs available to NLM applications to request memory services. Geist Jr.'s technique intercepts memory allocation calls made by a user program via the C library to the kernel, and logs the allocations and releases resulting from those application calls. The tracker logs the allocations and releases using two simple linked lists: an allocated block list and a message list. The technique generates a report on allocations without release and release requests for unallocated

memory. The tracker works by modifying the entry points of the APIs for memory allocation routines in the C library. A jump instruction is placed at the beginning of the allocation routines to an assembly language routine that calls the rest of the original routine and then calls a stub at the end to log allocation or deallocation requests made. Geist Jr.'s allocation tracker aids developers of application programs find memory allocation problems in their application programs.

Like Geist Jr., Donnelly relates to tracking memory allocation and deallocation for an application program to detect memory leaks. The memory management functions of the operating system are replaced with a set of debugging memory management functions. Unlike applicant's verifier that operates in kernel mode, Donnelly's technique uses debugging memory management functions to track memory allocations and deallocations for an application program that operates in user memory space. The set of debugging memory management functions extends the memory management functions of the operating system to pass expiration events and dependent pointers to the memory allocation table indicating when a corresponding memory allocation should be de-allocated. Expiration events are occurrences that indicate corresponding memory allocations should be de-allocated prior to the occurrence of the event. Dependent pointers to currently allocated blocks of computer memory indicate that the corresponding computer memory allocation should be de-allocated prior to deallocation of the computer memory allocation referred to by the dependent pointer. Once the expiration event and/or dependent pointer are specified in a memory allocation table, memory leaks can be detected by invoking debugging memory management functions from an application program.

Turning to independent claims 1, it recites among other limitations "receiving a request from a kernel mode driver," "determining that the kernel mode driver is to be monitored," "re-vectoring the request to a kernel mode driver verifier," and "taking action in the kernel mode driver verifier to actively test the kernel mode driver for errors."

Whenever a particular driver is to be monitored in applicant's technique, a kernel system call received from the driver is redirected to the driver verifier. The driver verifier invokes a kernel test function in place of the requested kernel function. Both the driver and the driver verifier are kernel components executing in the kernel address space. Geist Jr. does not teach receiving a request from a driver, nor does Geist Jr. teach re-vectoring the request received from a driver to a driver verifier as alleged. Rather the section of Geist Jr. that are referenced describe the allocation tracker that operates in user memory space. Geist Jr. describes replacing the C library memory function calls in the application executable with a jump instruction to a routine that records memory allocations and deallocations before executing the original API invoking the requested memory function. Such C library functions are linked and loaded in the user address space. The routine that Geist Jr. describes for recording memory allocations and deallocations is also part of the application executable loaded in user address space. When the C library function calls the kernel function, parameters are passed to the kernel function which executes in protected kernel space. Results are place in the user space for the user application to retrieve upon resuming execution. The modified C library function of Geist Jr. executes in user space and does not have direct access to kernel routines. Thus, Geist's tracker which executes in user space does not receive requests from a kernel mode driver. Nowhere in Geist is there any disclose of receiving a request from a kernel mode driver.

Nor does Geist Jr. teach re-vectoring the request received from a driver to a kernel mode driver verifier as alleged. The modified C library function of Geist Jr. executes in user space and does not have direct access to kernel routines. It intercepts application program memory allocation calls made by a user program via the C library to the kernel, and logs the allocations and releases resulting from those application calls.

Furthermore, Geist Jr. does not teach taking action in the kernel mode driver verifier to actively test the kernel mode driver. Geist Jr. does not have a driver verifier, only an application program tracker that logs allocation and deallocation requests. The tracker works by modifying the entry points of the APIs for memory allocation routines in the C library. A jump instruction is placed at the beginning of the allocation routines to an assembly language routine that calls the rest of the original routine and then calls a stub at the end to log allocation or deallocation requests made. The section of Geist Jr. referenced describes value-checking of parameters passed in the API call by the application program. Nor does Donnelly teach taking action in the kernel mode driver verifier to actively test the kernel mode driver. Donnelly's technique uses debugging memory management functions to track memory allocations and deallocations for an application program that operates in user memory space. Device driver interfaces execute within the kernel memory space and are not accessible by application programs executing in user memory space. Unlike Geist Jr. and Donnelly, applicant's technique instead provides separate kernel functions which may replace existing kernel functions. Applicant's substitute kernel functions execute in kernel memory space and have access to kernel routines and may receive device driver requests. Applicant's driver verifier performs automatic verification of a device driver by applying test conditions designed to

detect specific errors during controlled operation of the executing driver. For example, applicant's driver verifier can specifically test for the driver accessing memory locations outside of those memory locations allocated to the driver. This is accomplished by allocating the driver's memory from a special pool owned by the driver verifier which it has explicitly bounded by inaccessible memory space. The driver verifier also tests a driver for attempted access to deallocated memory by marking deallocated memory from the verifier's special pool as inaccessible when it is deallocated. The driver verifier also can test a driver for incorrect usage of a spinlock by controlling memory available to a driver under test. This is accomplished by randomly failing requests from the driver for additional allocation of memory. Applicant's technique specifies what errors to test for and performs test verification. Neither Geist Jr. nor Donnelly describe testing of device drivers because the application tracker and application programs cannot access device driver interfaces and can not examine resource lists for undeleted timers, pending deferred procedure calls, undeleted lookaside lists, undeleted worker threads, undeleted queues and other similar kernel resources. Furthermore, Parker does not disclose taking action in the kernel mode driver verifier to actively test the kernel mode driver. Claims 2-16, by similar analysis, are not rendered obvious by the relied-upon references.

Applicant respectfully submits that claims 27-30 are not rendered obvious as alleged because the relied-upon sections were incorrectly interpreted. Claim 27 recites among other limitations "restricting access to areas bounding the location" of allocated memory and "monitoring the areas bounding the location for an access violation using a kernel component." Geist Jr. and Donnelly do not teach restricting access to areas bounding the location using a kernel component. The section of Donnelly relied-upon

describes a debugging function for passing an expiration event to indicate corresponding memory allocations should be de-allocated prior to the occurrence of the event. Nowhere in Donnelly is there any description of restricting access to areas bounding the location of allocated memory and monitoring the areas bounding the location for access violation. Unlike Donnelly's application programs, drivers are kernel components executing in the kernel address space. Device drivers may operate in kernel memory space without the protection of the operating system and can crash the entire system with a stray pointer access. A stray pointer access means accessing memory locations outside of the blocks of memory allocated to the driver. Neither Geist nor Donnelly suggest or imply that the area of memory before the allocated memory block is restricted from a device driver. Nor does Geist Jr. or Donnelly suggest or imply that the area of memory after the allocated memory block is restricted from a device driver. However, applicant's driver verifier can specifically test for the driver accessing memory locations outside of those memory locations allocated to the driver. This is accomplished by allocating the driver's memory from a special pool owned by the driver verifier which it has explicitly bounded by inaccessible memory space. Claims 28-30, by similar analysis, are not rendered obvious by the relied-upon references.

Applicant respectfully submits that claim 31 is not anticipated by Geist Jr. because the relied-upon sections were misinterpreted. Claim 31 recites "receiving a plurality of requests from a kernel mode driver for allocation of various distinct sets of memory" and "determining from the tracking whether memory remains allocated to the driver at a time when the driver should have no memory allocated thereto." Geist Jr. does not describe this as alleged. The section of Geist Jr. relied-upon describes a use of his application

tracker to see where unfreed memory in an application was originally allocated. Geist does not suggest or describe that his application tracker is able to determine whether a driver has failed to free memory allocated by another component as alleged. Unlike Geist Jr.'s application programs, drivers are kernel components executing in the kernel address space and are not accessible by application programs executing in user memory space. Applicant's technique detects when a driver unloads without deallocating unneeded memory which was originally allocated by another component. Geist Jr. does not teach this as alleged.

Applicant respectfully submits that claim 32 is not rendered obvious by Geist Jr. and Donnelly because the relied-upon sections were misinterpreted. Claim 32 recites "receiving information corresponding to a kernel mode driver unload" and "determining whether resources remain associated with the driver," and "if so, generating an error." Neither Geist Jr. nor Donnelly describe or suggest detecting unreleased resources and generating an error upon driver unload. Applicant's verifier checks that any requested resources are deleted when the driver unloads. These resources include queues, lookaside lists, worker threads, timers and other resources. Geist Jr. describes an application monitor that operates in user memory space and cannot access kernel components like the driver that executes in kernel memory space. Geist's tracker logs the allocations and releases for application programs using two simple linked lists: an allocated block list and a message list. The section of Geist Jr. relied-upon describes the data structure for the message list that includes an error code for application programs errors. Geist Jr. does not describe generating errors for resources that are unreleased by kernel-mode drivers. The section of Donnelly relied-upon instead describes a debugging

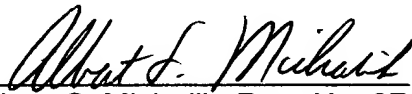
function for passing an expiration event to indicate corresponding memory allocations should be de-allocated prior to the occurrence of the event. Nowhere in Donnelly is there any description of detecting unreleased resources and generating an error upon kernel-mode driver unload. Claims 33-34, by similar analysis, are not rendered obvious by the relied-upon references.

Applicant respectfully submits that claims 35-37, by similar analysis as claim 1, are not anticipated by Geist Jr. because the relied-upon sections were misinterpreted. Moreover, claim 38 is not rendered obvious by Geist Jr., Donnelly and Parker because the relied-upon sections were misinterpreted. Furthermore, claims 39-52, by similar analysis as claim 1, are not rendered obvious by Geist Jr. and Donnelly because the relied-upon sections were misinterpreted. Claims 35-47 include the limitations of a "re-vectoring component" and a "driver verifier component." As discussed above regarding claim 1, Neither Geist nor Donnelly explicitly teach re-vectoring the request received from a driver to a driver verifier. Neither Geist Jr. nor Donnelly describe providing a test specification and perform automatic test verification like applicant's driver verifier component. Furthermore, neither does Parker describe a "re-vectoring component" and a "driver verifier component" regarding claim 38. Claims 48-52 include the limitations of "selecting one or more tests for verifying functionality of a system component" and "modifying a request for system services to include execution of the selected tests." Again, Geist Jr. and Donnelly do not provide test specification or perform automatic test verification, as does applicant's driver verifier component.

CONCLUSION

Based upon the above remarks, all of the pending claims are in condition for allowance. Applicants respectfully request reconsideration of this application and its early allowance. If in the opinion of the Examiner a telephone conference would expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney at (425) 836-3030.

Respectfully submitted,



Albert S. Michalik, Reg. No. 37,395
Attorney for Applicants
Law Offices of Albert S. Michalik, PLLC
704 - 228th Avenue NE, Suite 193
Sammamish, WA 98074
(425) 836-3030
(425) 836-8957 (facsimile)

In re application of WANG, LANDY
Serial No. 09/447,501

CERTIFICATE OF MAILING

I hereby certify that this Amendment and Petition for Extension of Time, along with Transmittal are being deposited with the United States Postal Service on the date shown below with sufficient postage as First Class Mail in an envelope addressed to:

Commissioner for Patents, Alexandria, VA 22313-1450.

Date: June 29, 2004



Albert S. Michalik

2660 Third Amendment